WHAT IS CLAIMED IS:

1. A processor comprising:
a register file including a plurality of registers; and
a functional unit coupled to the register file, the functional unit that executes an instruction operating upon a plurality of registers in the register file, the instruction in which a register specifier is implicitly derived, based on another register specifier.

2. A processor according to Claim 1 further comprising:
a decoder coupled to the functional unit and coupled to the register file, the decoder implicitly deriving a register specifier based on an explicitly-specified register specifier of the instruction.

3. A processor according to Claim 1 wherein:
the register specifier is implicitly derived by adding one to an explicitly-defined register specifier.

4. A processor according to Claim 1 wherein the processor is a Very Long Instruction Word (VLIW) and further comprising:
a register file including a plurality of register file segments;
a plurality of functional units, ones of the plurality of functional units being coupled to and associated with respective ones of the register file segments.

5. A processor according to Claim 1 wherein:
a multiply-add instruction uses an implicitly-derived register specifier and has a form of:

muladd rs1, rs2, rd,

and performs an operation specified by the equation:

$$rd = (rs1 * [rs1+1]) + rs2,$$

where the term [rs1+1] designates data contained within the register following the explicitly-defined register rs1.

6. A processor according to Claim 1 wherein:
the instruction is that
a bit extract instruction uses an implicitly-derived register specifiers and has a form of:

bitext rs1, rs2, rd,

and performs an operation of extracting bits from even-aligned pairs of registers r[rs1] and [rs1+1] where the term [rs1+1] designates data contained within the register following the explicitly-defined register rs1.

7. A processor according to Claim 1 wherein:
the instruction is that
a call instruction uses an implicitly-derived register specifiers and has a form of:

call label,

causing a control transfer to an address specified by a label operand, the address of the instruction word following the instruction word begun with the call instruction is held in an alias register, an assembler using an alias link pointer lp for the alias register.

8. A processor according to Claim 1 wherein:
the instruction is that
a double-precision floating point add instruction uses an implicitly-derived register specifiers and has a form of:

dadd rs1, rs2, rd,

and performs an operation specified by the equation:

(rd, [rd+1] = (rs1, [rs1+1]) + (rs2, [rs2+1]),

where the terms (rs1, [rs1+1]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double-precision words.

9. A processor according to Claim 1 wherein:
a double-precision floating point compare instruction uses an implicitly-derived register specifiers and has a form of:

dcmpcc rs1, rs2, rd,

and performs an operation of comparing data in registers (rs1, [rs1+1]) with data in registers (rs2, [rs2+1]) and storing a result in registers (rd,[rd+1]) where the terms (rs1, [rs1+1]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double-precision words, and cc designates a condition code including equal, less than, and less than or equal to conditions.

10. A processor according to Claim 1 wherein:
a double-precision floating point multiply instruction uses an implicitly-derived register specifiers and has a form of:

dmul rs1, rs2, rd,

and performs an operation specified by the equation:

(rd, [rd+1] = (rs1, [rs1+1]) * (rs2, [rs2+1]),

where the terms (rs1, [rs1+1]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double-precision words.

11. A processor according to Claim 1 wherein:
a double-precision floating point subtraction instruction uses an implicitly-derived register specifiers and has a form of:

dsub rs1, rs2, rd,

and performs an operation specified by the equation:

(rd, [rd+1] = (rs1, [rs1+1]) - (rs2, [rs2+1]),

where the terms (rs1, [rs1+1]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double-precision words.

12. A processor according to Claim 1 wherein:
a pack instruction uses an implicitly-derived register specifiers and has a form of:

pack rs1, rs2, rd,

and operates upon a register pair (rs1, [rs1+1]) as four signed 16-bit operands, and shifts the four operands right by a value designated by the register specified by rs2, clips the shifted 16-bit operands within defined limits and stores the clipped 16-bit operands in a register pair (rd, [rd+1]).

13. A processor according to Claim 1 wherein:
a double-precision floating point conversion instruction uses an implicitly-derived register specifiers and has a form of:

dtox rs1, rd,

and performs an operation of converting a double-precision floating point value to a specified format x, the format x including a single-precision floating point format (dtof), an integer format (dtoi), and a long integer format (dtol).

14. A processor according to Claim 1 wherein:
a double-precision floating point absolute value instruction uses an implicitly-derived register specifiers and has a form of:

dabs rs1, rd,

and performs an operation of converting a double-precision floating point value in a register pair (rs1, [rs1+1]) to an absolute magnitude in a register pair (rd, [rd+1]).

15. A processor according to Claim 1 wherein:
a double-precision floating point negative value instruction uses an implicitly-derived register specifiers and has a form of:

dneg rs1, rd,

and performs an operation of converting a double-precision floating point value in a register pair (rs1, [rs1+1]) to a negative magnitude in a register pair (rd, [rd+1]).

16. A processor according to Claim 1 wherein:

a double-precision floating point set limit instruction uses an implicitly-derived register specifiers and has a form of:

dlim rs1, rs2, rd,

and performs an operation of setting a double-precision destination register (rd, [rd+1]) to the maximum of a double-precision first source register (rs1, [rs1+1]) and a double-precision second source register (rs2, [rs2+1]), or setting the double-precision destination register (rd, [rd+1]) to the minimum of a double-precision first source register (rs1, [rs1+1]) and a double-precision second source register (rs2, [rs2+1]),

where the terms (rs1, [rs1+1]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double-precision words.

17. A processor according to Claim 1 further comprising:

a decoder coupled to the functional unit and coupled to the register file, the decoder implicitly deriving a register specifier based on an explicitly-specified register specifier of the instruction and generating a first pointer to the explicitly-specified register and a second pointer to the implicitly-derived register.

18. A processor according to Claim 1 further comprising:

a decoder coupled to the functional unit and coupled to the register file, the decoder implicitly deriving a register specifier based on an explicitly-specified register specifier of the instruction and generating a first pointer to the explicitly-specified register and a second pointer to the implicitly-derived register.

19. A processor according to Claim 1 further comprising:
a pointer coupled to the register file and designating a register in the register file, the pointer including a signal indicative of selection of a implicitly-derived register, the register file generating two pointers, one directed to the explicitly-specified register and a second directed to the implicitly-derived register when implicit derivation of a register specifier is selected.

20. A method of operating a processor comprising:
storing information in a register file including a plurality of registers;
executing instructions in a functional unit coupled to the register file and operating upon a plurality of registers in the register file;
explicitly defining a register specifier of a register operated upon during executing of the instruction; and
implicitly deriving a register specifier based on the explicitly defined register specifier.

21. A method according to Claim 20 further comprising:
decoding an instruction; and
deriving, during decoding of the instruction, a register specifier based on an explicitly-specified register specifier of the instruction.

22. A method according to Claim 20 further comprising:
implicitly deriving the register specifier by adding one to an explicitly-defined register specifier.